International Journal of Child-Computer Interaction ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

International Journal of Child-Computer Interaction

journal homepage: www.elsevier.com/locate/ijcci

Playing Beowulf: Bridging computational thinking, arts and literature through game-making

Bruno Henrique de Paula [a,*], Andrew Burn [a,*], Richard Noss [a], José Armando Valente [b]

[a] UCL Knowledge Lab, UCL Institute of Education, University College London, London, UK
[b] Instituto de Artes—Universidade Estadual de Campinas, Campinas, Brazil

ARTICLE INFO

Article history:
Received 3 February 2017
Received in revised form 2 November 2017
Accepted 9 November 2017
Available online xxxx

Keywords:
Game-making
Computational thinking
Arts
Humanities

ABSTRACT

Preparing younger generations to engage meaningfully with digital technology is often seen as one of the goals of 21st century education. Jeanette Wing's seminal work on Computational Thinking (CT) is an important landmark for this goal: CT represents a way of thinking, a set of problem-solving skills which can be valuable when interacting with digital technologies, and with different fields of knowledge, such as Arts and Humanities. Even if this cross-areas relevance has been celebrated and acknowledged in theoretical research, there has been a lack of practical projects exploring these links between CT and non-STEM fields. This research develops these links. We present a specific case – a game produced by two 14 years-old boys – within *Playing Beowulf*, a collaboration with the British library's Young Researchers programme, in which students aged 13–14 from an inner-London (UK) school have developed games based on their own readings of the Anglo-Saxon poem Beowulf during an after-school club. The game was produced using *MissionMaker*, a software (currently under development at UCL Knowledge Lab) that allows users to create and code their own first-person 3D games in a simple way, using pre-made 3D assets, such as rooms, props, characters and weapons and a simplified programming language manipulated through drop-down lists. We argue that *MissionMaker*, by simplifying the development process (low floor), can be a means to foster the building of knowledge in both STEM (CT) and Arts and Humanities, building bridges between these two areas inside and outside traditional schooling.

1. Introduction

In recent decades, we have witnessed a great dissemination of the benefits of computer-related skills for Education. Notions such as *procedural thinking* [1], *procedural literacy* [2,3] or *computational literacy* [4] have fed into a valorisation of computational skills in schools in different countries. Jeanette Wing's [5] work is often recognised as one of the landmarks of this new movement: she envisaged her term "Computational Thinking" (CT) as a "formative skill" for the contemporary world, at least as important as literacy and numeracy [6], essential for citizens' lives in an era in which digital technologies are ubiquitous.

The importance of this "formative skill" is clear in different domains. In media, for instance, we can directly relate it to what Manovich [7] names "new media" (digital media). These are constituted by two different layers: the 'cultural layer', which refers to cultural forms – the "language" – of media (e.g. film, television, videogames and, recently, the 'convergence culture' [8]), and the 'computer layer', referring to the information-processing aspects of "new media" [7], which modify how the "language" of media functions. In "new media", meaning emerges from this confluence between these two layers.

However, even if the cultural and computer layers are inextricably entangled, conceptions of media literacy in recent years [9,10] have usually focused on the cultural aspect. By contrast, conceptions of computational thinking have focused on the computer aspect, not necessarily emphasising it as a social practice with deep cultural and communicational impacts [11]. Therefore, we need to re-examine how people think about culture and computing, both in relation to the (media) arts, and in relation to computing.

In this paper, we focus in how CT, as a "formative skill", can help to establish bonds between computing-related concepts and the conceptual frameworks required by the media and the arts. We focus on a specific approach: digital game-making. We present a case study from a project in which a different perspective towards game-making and formal learning was employed: by using the Anglo-Saxon poem *Beowulf* as the basis for the project, we aimed at understanding how CT and Arts and Humanities (AH) can be fostered simultaneously in an educational context.

The main results and further discussions related to this case study will be presented in this paper in Section 4. Prior to that,

* Corresponding authors.
E-mail addresses: bruno.paula.15@ucl.ac.uk (B.H. de Paula), a.burn@ucl.ac.uk (A. Burn).

https://doi.org/10.1016/j.ijcci.2017.11.003

Section 2 introduces a brief literature review on relevant aspects regarding CT and on the theoretical foundations that guided our work on bridging CT and AH. Additionally, Section 3 describes the tool employed in our project – *MissionMaker* – as well as the main methods used to structure and analyse our data.

2. Context

Jeanette Wing's works [5,12] are often seen as landmarks for a new phase on computers and computer-related skills in schools. Since then, CT is emerging as a desired learning goal in different educational systems, even becoming part of official educational policies in some countries [13,14]. This does not mean, however, that CT can be considered a stable, closed term [15], and several definitions have emerged trying to pin down its core concepts [5,12,16]. Nevertheless, there is some consensus about foundational aspects such as abstraction, automation and reduction; and, more importantly, that computational thinking refers to a set of problem-solving skills based on computer-related concepts [17].

But how have different countries incorporated CT into educational systems? One of the current trends among policy-makers is to use activities based on coding and programming to foster CT. Some of these initiatives include "Computer Science is for everyone" [18], an effort from the US White House to promote and popularise Computer Science (CS), and the new National Curriculum in England, which has CS as one of its bases [13].

While CT can certainly be fostered through this path, and some authors [19] argue that programming is essential to engage with CT's common practices, CT cannot be understood as a synonym – or even an "abridged version" – of CS. While the latter is recognised as an academic discipline that studies computers and computational systems, the former is understood as a specific set of thought processes used to solve complex problems [20]. In this sense, CT is not – and should not be taught as if it were – necessarily bound to CS, since it can be used as a general tool in different domains and not only in those that are part of CS:

> the ultimate goal [of CT] is not to teach everyone to think like a computer scientist, but rather to teach them to apply these common elements to solve problems and discover new questions that can be explored within and across all disciplines [21].

Some initiatives aimed to foster CT in this broader sense, exploring its alleged cross-curricular potential through different scenarios, such as games, journalism, science and engineering [22]. Other research also presented an array of activities with potential to embody this broader view on CT, such as the Computer Science Unplugged, where students learned computational concepts without relying on digital technologies, or initiatives exploring robotics, digital narratives, simulations and games [23–25].

Producing games is arguably one of the most consolidated approaches to deliver CT in educational contexts [21,26], mainly because it provides an opportunity to connect several "formative skills" within the same task, while also stimulating students' creativity and engagement with diverse modes of communicating meanings. Another argument in favour of games when aiming at CT is their systemic nature [27]: games are often condemned and considered frivolous [28], but we cannot ignore that, in many cases, they are some of the first experiences that we have of formal systems, fostering the understanding of correlations between rules and outcomes and of how to operate these rules.

However, even if this possibility of cross-curricular work through games is celebrated throughout the literature [29,30], the majority of practical work in this field is still bound to STEM-related academic content or to the sole development of programming principles [26], while there is a shortage of initiatives trying to establish stronger bonds with other parts of the curriculum, especially with AH.

One path to bridge this gap and connect CT and AH is through Proceduralism [31–33]. Murray [33] argues that one of the essential properties of digital environments is procedurality, which allows them to execute a series of rules and, consequently, opens an opportunity for "encapsulating specific real-world behaviours into programmatic representations" [31, p. 13]. In other terms, this capacity of creating loops and specific logic-oriented sequences modifies how we can use media to express meaning.

This capacity is further explored by Bogost [32] under the scope of procedural rhetoric: the idea that arguments can be established and used to persuade an audience through processes in any kind of cultural production: "any medium—poetic, literary, cinematic, computational—can be read as a configurative system, an arrangement of discrete, interlocking units of expressive meaning" [32, p. ix].

Procedural rhetoric sees in processes (especially computational ones, such as game rules) the ability to frame situations and to express meanings to an audience (players, spectators, readers). It signals the importance of being able to read systems and digital processes not only in a functional way (e.g. how to give specific instructions to a computer to make a character jump in a game), but also to understand what arguments might be implicit in these processes (e.g. what does it mean to give the ability to the character to jump in a game? Can the character run and jump at the same time? And what is the narrative significance of jumping?) [32]. Examples of this application can range from this silly "jumping" routine to more complex (and maybe, more controversial) ones, such as an hypothetical online RPG games in which avatars have genders and can have affective relationships among themselves, but only if they are from different genders. In this sense, it is clear that the game – through its code – is giving a powerful message about same-sex relationships.

Proceduralism, in some sense, highlights the importance of thinking about how processes (computational or not) can be interconnected and communicate meaning; this comprehension can be seen as one of the first steps for the problem-solving heuristic defended by CT, and are part of what Bogost defines as *procedural literacy* – "[...] the ability to reconfigure basic concepts and rules to understand and solve problems, not just on the computer, but in general" [2, p. 32].

Proceduralism as a current of thought is often criticised for some of its positioning: Sicart, for instance, considers that it favours the designer/writer's ideas rather than acknowledging the value of other possible personal interpretations by the audience [34]. While this critique is relevant – and we should bear it in mind while approaching works from a proceduralist perspective – it emphasises how computer-related processes are not necessarily neutral, but can carry values and communicate meanings in different ways. The proceduralist argument reminds us that, after all, CS and AH are not so far apart as they might seem at first sight: meaning is produced in all cases, and although the meaning carried by a narrative might seem more evident in a first moment, code and computational processes also generate meaning and demand interpretation; thus, when working within this domain we should be aware of what is being communicated by our production. Proceduralism, however, is not the only way to approach CS and AH.

Other research successfully explored narratives as a means to help students deal with CT-related concepts, such as abstraction. Here, we understand abstraction as a kind of generalisation by removing detail from a complex object or process, in order to construct general concepts that might be used in other objects/processes [35]. Mathematics, due to its nature, has in abstraction one of its core elements and often pupils struggle to move from its abstract forms to more concrete ones. Mor and

Please cite this article in press as: B.H. de Paula, et al., Playing Beowulf: Bridging computational thinking, arts and literature through game-making, International Journal of Child-Computer Interaction (2017), https://doi.org/10.1016/j.ijcci.2017.11.003.

Noss [36] describe how narratives can work as an epistemic vehicle to exemplify situated abstraction, meaning that narratives can help pupils to understand better how abstraction works by offering some context in which abstract concepts can be applied (and abstraction is demanded).

One of the most interesting outcomes of their research was the reinforcement of the role of programming as a mediating form between narratives and formal, unambiguous (and abstract) mathematical language:

> We see programming as an expressive activity, a form of writing or composing, contingent on context and used purposefully to carry out actions. [...] Programming can afford a narrative form for representing mathematical meanings [36, p. 215].

An important distinction here is that programming skills were not the ends of the initiative, but the means, combined with narratives, to achieve another specific goal – a better comprehension of (Mathematical) abstraction.

Furthermore, it can be argued that these narratives and CT principles share some essential aspects: for instance, narratives can be procedural in certain ways. They can depend on rule-governed behaviour with a string of dependencies; they can be in a state of constant conditionality; they can rely on predictable and repeatable forms of behaviour. The kinds of narrative conforming to this sort of procedurality are the narratives of popular fiction, which depend on recognisable formulaic constructs (such as superheroes with familiar costumes, powers, dual identities); of folk tale, which is constructed on the oral-formulaic principles [37,38] of high redundancy, familiar character types, and modular narrative elements [39]; and more broadly, archaic narratives also conforming to these principles. These principles also characterise computer games, the difference being that the formulaic structures manipulated by the poet/performer in oral narrative are deployed by computer programmes: game engines, animations, character modules and so on [40]. One reason for the choice of Beowulf in this project is that it is the ideal example of the oral-formulaic narrative. Beowulf, through its "algorithmical loops" – e.g. Beowulf fighting different monsters – can be understood as a good example of how non-digital environments can embody procedural aspects [31].

We can argue, then, that literature and games are closely-related cultural forms, not just because of narrative content such as mediaeval fantasy, but because the very grammar of these narratives and game programming are similar in certain ways: procedurality and the computational thinking it involves –they all carry meanings and demand interpretation, and they also demand problem-solving, working backwards from projected outcome to cause and condition, designing rules to govern the logic of the imaginary world, its characters and events. To pose such analogies is to productively challenge both STEM/computing and the arts. How might students (or anyone) *program* literature? Poetry? Drama? Shakespeare? [41]. What might it mean to solve problems of narrative sequence, of character motivation, of dramatic action? And conversely, for the arts – how could we look at these art-forms afresh if we considered the functions of rule, formula, economy, quantification in narrative, dramatic speech and gesture, human behaviour?

A further argument comes from multimodality theory [42]. Games are multimodal ensembles integrating the semiotic modes of moving image, music, spoken and written language, all governed by the orchestrating function of computer programs, which can be seen as a kind of meta-mode. This notion of "code as mode" allows us to explore not only how students learn to think in the way it requires, but also how they must understand the structures, aesthetics and affordances of the other semiotic modes in play [43]. It requires an expanded notion of code, for students to understand not only its programming functions, but also what Marion Walton [44] describes as "the level of the procedural rhetoric, aesthetics and poetics encoded in a work".

Games, due to their procedural nature and their reliance on narratives, seem, in theory, the perfect conduit for bridging CT, CS and AH in an educational context. But how can we operationalise this connection between these different spheres of the curriculum? Are students able to deal simultaneously with different modes of thinking (CT, CS and AH) when making games? Can CT-related concepts work as a bridge for linking CS and AH? Does complexity increase simultaneously in these different domains in game-making, or more complex narratives mean necessarily less complex rules/programming patterns? In order to answer these questions, we present a small exploratory case study from an after-class game-making club that was part of *Playing Beowulf* project.

3. Project context and methods

3.1. Context and summary of the project

Playing Beowulf was an Arts and Humanities Research Council (AHRC) funded project which aimed to promote further engagement with the epic Anglo-Saxon poem *Beowulf* by bringing together a thousand-year-old text, digital technologies and new means of creative expression[1] through MissionMaker (described in Section 3.3).

Beowulf tells the story of the eponymous Scandinavian warrior who supports Hrothgar, king of the Danes, in defending his domains in Heorot against the monsters Grendel and Grendel's mother. After his successful campaign against these two monsters, Beowulf returns to his homeland and becomes king eventually. For fifty years the hero reigns, until a Dragon attacks his kingdom; he is able to defend his domains, slaying the Dragon, but is mortally wounded in the process and dies.

This text was chosen for this project for some specific reasons. Firstly, *Beowulf* is part of an influential tradition in digital games' culture: the setting and the kind of narrative presented by the poem are antecedents of games which explore this kind of Nordic–Medieval–Fantasy domain; indeed, such games, and the RPG genre in general, can be said to derive from the fiction of JRR Tolkien, which itself draws heavily on *Beowulf* and similar tales [45]. More importantly, as mentioned before, *Beowulf* presents an interesting algorithmical loop – monster appears, fight happens, reward – easing then the process of adapting a literary work to a videogame.

In this paper,[2] we will base our discussion on a specific case, a game produced by two boys in a specific module within this larger project, which was jointly organised by the British Library (BL) and the DARE centre (University College London Institute of Education/Knowledge Lab) in an inner-London school as part of the British Library's Young Researchers program. Two teachers from this school (English and ICT) also took part in the organisation and development of the activities.

The project was structured as an after-class program constituted of 6 one-hour voluntary after-class sessions from October to December 2015. Sessions occurred in two sites: the BL, where students saw the unique manuscript of the poem, revisited the story (since most of them had studied it in previous years) and started planning how it could be translated into a game; and the Learning Centre (LC) adjacent to the school. Here, students focused on the game production using *MissionMaker*, while researchers and

[1] See Section Acknowledgements for AHRC reference.

[2] Some of the results were presented and discussed on a different perspective – focusing on self-expression and identities rather than on approaches combining CS, CT and AH – elsewhere [46].

Please cite this article in press as: B.H. de Paula, et al., Playing Beowulf: Bridging computational thinking, arts and literature through game-making, International Journal of Child-Computer Interaction (2017), https://doi.org/10.1016/j.ijcci.2017.11.003.

teachers supervised their work, roaming through the class helping students. Since our main aims were related to the translation of the classical text into a game, we opted for an implicit approach for dealing with CS: concepts such as algorithms, types of data and Boolean logic were sometimes part of the discussions, but were neither highlighted nor reinforced. The sessions were organised to ensure that at least three researchers (one teacher, one researcher from the BL and one researcher from DARE or UCL) were present.

The two boys that produced the game analysed in this paper were both Year 9 students (13–14 years old) with good academic performance and no behavioural issues. There was a good relationship between them and with their peers and teachers, and both came from multicultural families, composed by British and Latin-American parents (in one case, Brazilian, and in the other, Argentinean). Our initial survey also indicated that they were experienced in videogames, something that was confirmed throughout the project, especially during the interviews.

3.2. Methods

This was an exploratory qualitative research loosely based in ethnographic methods, in which we followed the group of students throughout the process of making a game inspired by *Beowulf*, from the inception to the final steps in the game development using *MissionMaker*. We opted for following a case study approach, focusing specifically in the design process and the final product created by two students.

An initial survey was used to better understand students' experience of games. Apart from this initial survey, our main research instruments during the project sessions were three: observations, interviews – both during all sessions – and game analysis. The latter method can be considered the most important source for our research, and the former two were used mainly to triangulate our findings from game analysis.

Observations were carried out by the first author and used mostly to understand how students interacted with the software (e.g. how easy was to translate their design ideas into the software, how it limited/interfered in their design decisions) and with each other (e.g. how they gave feedback to each other about their ideas); they were registered through notes in a research journal kept by the researcher. The interviews were short conversations (video recorded) conducted by the first or second authors while students were working in their games during the sessions at the LC: in these cases, researchers used the games being developed to probe for understanding their design decisions and their knowledge about CT/CS. The main starting point for these interviews was to have students explaining about their games to the researchers, and then researchers explored further with different questions. The first author focused especially on the case presented here, working closely with the students.

After each session, a copy of the current version of their games was saved, and was then analysed immediately in relation to their main ideas for their game and to their achievement in terms of game design until that moment. Saving a working copy of students' games after each session also allowed us to track their progress during the whole initiative, both in terms of game design – understanding how their games were located in the broad spectre of game genres, if narratives were used and what kind of challenges were presented to the player – and of game computational sophistication [47] – how complex the rules made by the students were.

All the data generated throughout the sessions was analysed under a Multimodal framework [48]. By adopting a Multimodal-based approach, we were able to understand how the participants have used different modes (visual, aural, ludic, code, etc.) to make meaning. Some questions used to evaluate the game were: Who is the main character? What is his/her objective? Is the objective clear for the player? How does the player know her objectives? Are sound effects in the game?

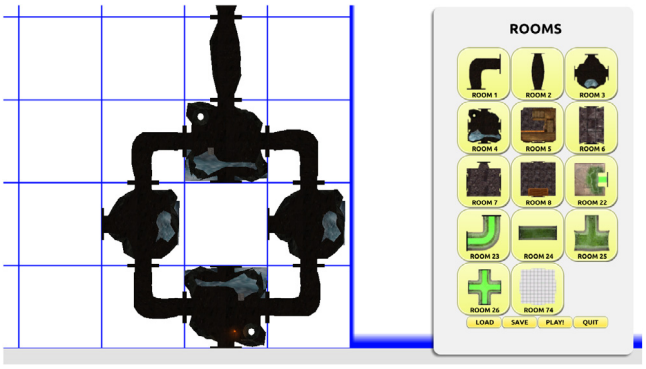

Fig. 1. Environment Mode, where the environment is designed.

Answers to these questions were produced by manipulating the participants' game (both playing and analysing it in Edit Mode, in order to examine the code) without their presence. These findings were then clarified with the participants in the interviews during the following week, in order to look for validity of our analyses through comparing it with participants' ideas and intentions.

3.3. A brief overview of MissionMaker

The software used by participants to produce their games during this project was an updated version of *MissionMaker*. Produced originally in 2007, it allows users to create their own 3D, first-person games without having to resort to complex knowledge of 3D modelling or computer programming; in some sense, *MissionMaker* works under the "powerful, yet simple" paradigm, since it enables users to produce complex products in a simple way.

The main dynamic of this software lies in selecting and organising ready-made available assets, establishing logical outcomes according to the actions taken in the game through the creation of rules. Its latest Unity-based version also includes a pack of Viking-themed assets developed especially for this project, in order to allow users to create Beowulf-themed games, and a context-sensitive system of dropdown-lists to organise the sequence of commands. Additionally, the software also incorporates some computational concepts (e.g. Boolean logic), enabling users to create multiple conditions and/or commands in the same rule. Figs. 1 to 3 below illustrate the dynamic of working with *MissionMaker*.[3]

Users can build the environment of their game by dragging different ready-made rooms into a grid, generating a top-viewed map of their game world (Fig. 1).

After generating the game world, the user can populate it using a range of diverse ready-made assets, such as props, characters and weapons (Fig. 2).

The dynamics of the game are established through rules. The rule editor is composed by a set of at least two lines: one condition and one command, which will be executed if the condition specified before is met. Rules can be more complex by having several conditions (combined through Boolean logic) and multiple commands. The "coding" happens through context-sensitive dropdown lists, which means that they have a hierarchy starting from left to right: the values available to be selected in the following lists depend (and are updated according to) on the values selected in the previous lists. Fig. 3 presents a very simple rule, usually the

[3] An explanation of the working dynamics in the original version of *MissionMaker* can be found at [49].

Please cite this article in press as: B.H. de Paula, et al., Playing Beowulf: Bridging computational thinking, arts and literature through game-making, International Journal of Child-Computer Interaction (2017), https://doi.org/10.1016/j.ijcci.2017.11.003.

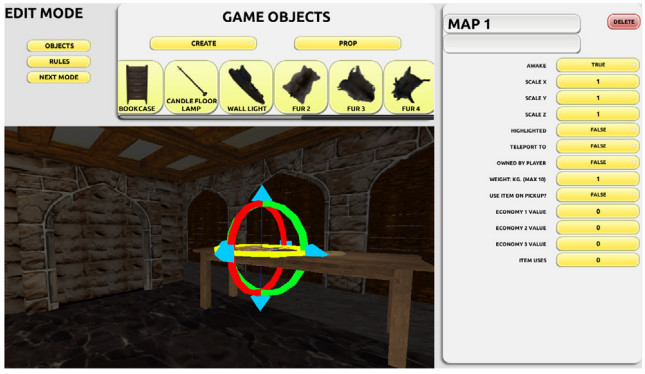

Fig. 2. Edit Mode, where entities are added to the game.

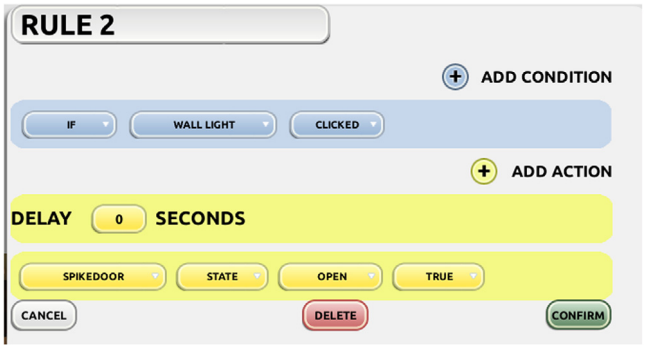

Fig. 3. A simple rule produced on the Rule Editor.

first made by users: a click in a specific object opens a specific door (technically, it manipulates the variable "Open" from the object named "SpikedDoor": it sets this Boolean variable value to "True"). More complex examples of rules shall be presented in the following sections of this paper.

One important aspect regarding "coding" in *MissionMaker* is the programming language: as noticed in Fig. 3, producers use a constrained form (context-sensitive lists) of natural language to design their games. This approach has already been used successfully in other research [50], providing an easier means to grasp how to organise and communicate commands to the computer than regular programming language; moreover,the constriction of natural language avoids syntax errors (e.g. typos or use of words not recognised by compilers). Due to this easier approach to game design, *MissionMaker* can be considered a "low floor" [51] tool, meaning that beginners can start producing their own games without having to resort to complex knowledge.

4. Rewriting Beowulf

In this paper, we focus on the game development process carried out by two 14-years-old boys working together during three sessions (totalising 3 h). Their game presented some level of complexity, and we identified two possible reasons for it: first, both of them presented a vast knowledge about games and its culture, and influences of some of their favourites – mainly blockbusters like Fallout or Skyrim – were noticed in their production. Second, they worked together in a collaborative way: they decided to separate the work into two domains ("narrative" and "mechanics"), but this separation did not prevent them from discussing and negotiating all design decisions, leading to an intensive iterative process (design, refine, reflect).

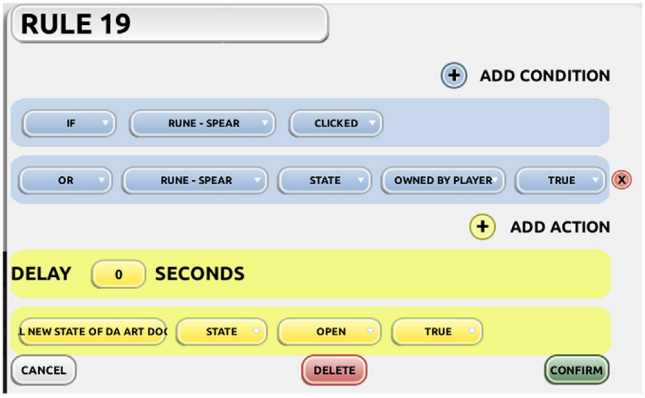

Fig. 4. Complex rule involving multiple conditions.

Computationally, their game presented patterns – coherent game mechanics [47] – and some complex features in coding, such as the use of Boolean operators (something that is not required by *MissionMaker*). Fig. 4, for instance, represents a rule found in their game, in which the command (the opening of one specific door) is triggered either if the player clicks the rune meaning "spear" **or** if the player collects it.

This example illustrates one of the main mechanics found in their game: clicking on "random" objects (such as a barrel) can lead to progress, rewards or even more challenges. This mechanic can be seen as reasonably simplistic, but is a pattern found throughout their game, creating then a coherent experience (even if difficult to understand at a first sight).

It is also interesting to notice how, in their speech, this mechanic becomes a kind of abstraction. In different occasions, both pupils called this click-outcome dynamic a "secret". The following excerpt is an example of this process:

> *Student 1: By giving rewards on the kind of places you'd have to click... So, there's one* ***secret*** *where, in the beginning, you have to make the king spawn in order to through the door, or else the guards will attack you, 'cause you're trying to go through them. We made it so you're supposed to click a barrel, but no person will just click a barrel at random, so we made that, so, there's a sword there, so, if you miss the sword, you'll click the barrel, and when you pick up the sword and you turn around, the king will be there.*

This use of a single term to denote the diverse instances of the same action can be interpreted as an indication of the "secret" as an abstraction, a general function that, in a regular programming language, could receive some parameters (what is to be clicked, what is the object affected, which properties of the affected object are altered) and produce a specific outcome. Since *MissionMaker* rule editor relies on a strict event-based programming system, the implementation of this abstraction (e.g. creating a custom function) is not possible.

More interesting, however, is that the "secrets" from their game did not only play a role as a proxy for a computational concept (an example of abstraction), but were also used in conjunction with narrative events. One clear example was the introduction of the Queen as a hidden character: she was only accessible through another "secret" (opening a hidden door through a click and clicking in a specific floor mat). Figs. 5 to 9 below illustrate the whole ingame process of summoning the Queen, as well as the rules that govern this process.

Her appearance in the game, however, is not only a mere secret; it modifies the game's dynamics, leading the player to a more difficult path (with more enemies to be defeated), while also offering

Please cite this article in press as: B.H. de Paula, et al., Playing Beowulf: Bridging computational thinking, arts and literature through game-making, International Journal of Child-Computer Interaction (2017), https://doi.org/10.1016/j.ijcci.2017.11.003.

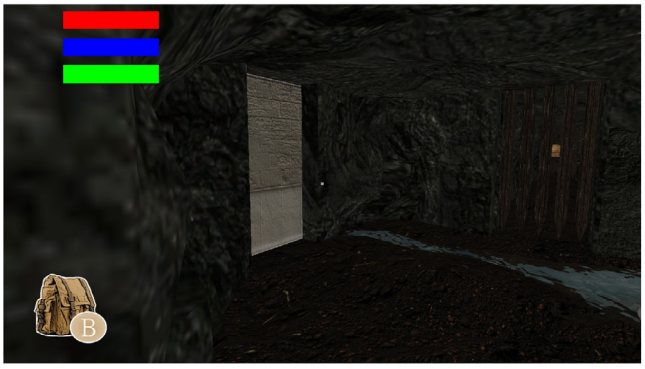

Fig. 5. The door (left) which protects the Queen room.

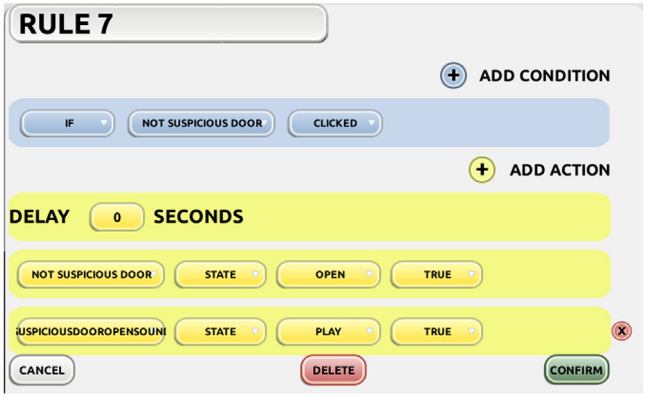

Fig. 6. Rule that opens the Queen's room door.

Fig. 7. Floor mat that should be clicked to spawn the Queen.

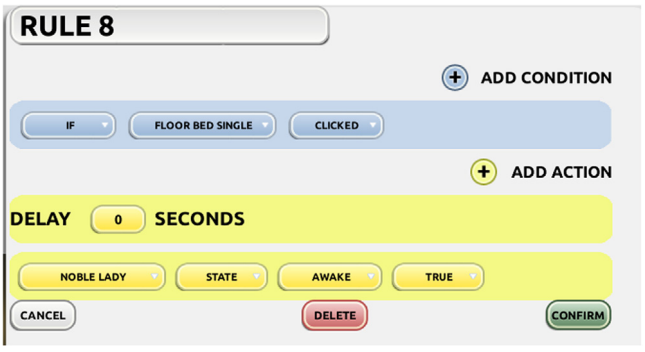

Fig. 8. Rule that spawns the Queen.

Fig. 9. The Queen is summoned after click on the floor mat.

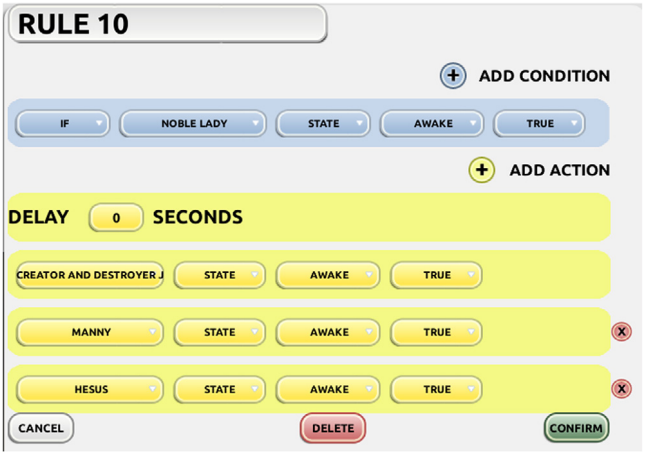

Fig. 10. Rule that spawns different (and more powerful) enemies if the Queen is summoned.

a bigger reward after the final battle. Fig. 10 below presents one of the rules related to this process, showing how more enemies are summoned if the Queen is found in the game.

This change in their game's dynamics can also be connected to narrative elements: in this case, these ludic changes were related to a supposed ambiguous position of the Queen, jealous about Beowulf's success but also fearful about the fate of her Kingdom, which was being attacked. This narratological justification for the Queen's position is found in the students' explanation about their game:

> *Student 2: It's kind of the beginning of the Beowulf poem where she kind of doesn't like Beowulf, because she wants her son to be king, and not Beowulf....*

Therefore, the reason why the Queen is a "secret" in the game is not only meaningful in terms of gameplay, but also regarding narrative aspects (she could not be seen undermining the King publicly, but at the same time she did not want Beowulf to succeed completely and become the following King to the detriment of her son). This secret is also well incorporated into their game, recruiting different modes: code (the rules presented above), visual (the presence of the Queen itself as a character) and ludic (the new – and more difficult – path that becomes available for players), creating then an interesting device to the progress of the game.

What is important here is that, even if the Queen's presence is narratologically meaningful – as justified by Student 2's speech above – this justification is never explicitly presented to the player via other meaning-making devices rather than the game's rules – in fact, the code. This use of the Queen as a narrative and ludic

Please cite this article in press as: B.H. de Paula, et al., Playing Beowulf: Bridging computational thinking, arts and literature through game-making, International Journal of Child-Computer Interaction (2017), https://doi.org/10.1016/j.ijcci.2017.11.003.

device, thus, can be understood as a rudimentary use of procedural rhetoric: the Queen does not want Beowulf to succeed, but this is (only) expressed in game via code (since no other mode is recruited in-game to expose this fact). In other words, the only way for the player to understand this problematic relationship between the Queen and Beowulf is through interacting and interpreting the game's code.

This example also highlights how the narrative itself can embody procedural aspects: it is often based on events (which can be treated as rules), and depending on different conditional tests (e.g. is Beowulf going to find the Queen? Is he going to slay the Dragon?), which leads to different outcomes (e.g. more and stronger enemies appear; he wins a reward if the Dragon is slayed).

At last, the "secret", by working simultaneously in two different domains (as a higher CS structure and also a narrative device), embodies the sort of mediation device that programming can provide between the "rigid", unambiguous form of a formal language (that of CS and Mathematics, for instance), and the more "fluid" and open language of AH (exemplified by narratives). What is interesting in this case that this bridge between the different domains did not only occur practically – *ingame* – but also terminologically: even if they approached their design process in two diverse halves ("mechanics" and "narrative"), they have found a common language to deal with game elements from different perspectives, represented here as "secret", acting then as a common space for the domains of CT, CS and AH.

In this sense, we cannot ignore the role that programming can play to bridge these fields. In this project, participants were effectively *programming* Beowulf – redesigning a literary narrative as a game by constructing rules organised through Boolean logic. In a general sense, it is possible to see analogies between the programming of events, rules and economies in a game and constructing a narrative. The characters, events, objects and locations in narrative are in a sense media databases, governed by a narrative algorithm. A rule constructed in *MissionMaker* reads: "IF Noble Lady state awake True/THEN Manny state awake True". This sentence is both a programmed rule AND a narrative declaration, which could be paraphrased as "The enemy named Manny enters the scene if the Queen also enters the scene".

Programming works, then, as a sort of mediator between these two domains (CT and AH, here represented by narratives). The code can be understood as an orchestrating mode [52], which organises and presents the interactions between different modes – moving images, sound, gameplay – in a unique context, establishing a coherent experience for both designers and players. This does not mean, however, that any kind of programming can play this role easily: arguably, the bridge is easier to build if the programming language is closer to natural language. This becomes clearer in the example of the rule/narrative cited above, and is consistent with other research, which showed that a programming language based in a constrained version of natural language bring better results (in terms of students' productions) when compared to "pure" programming languages or unconstrained natural languages used as codes [50].

5. Conclusions

Although this was a limited exploratory case study, we believe that interesting elements rise from our analysis. It presented some evidence of a possible intrinsic connection between the complexity level of narratives and computational aspects: the case presented showed – if we consider the time constrains – reasonably complex computing structures and narrative. This can be interpreted as a sign of how narrative and computational aspects can be connected and exemplifies how they can be operationalised simultaneously by students in educational contexts.

The example of the "secret" – which was meaningful both as an abstraction (a generalisation, making an idea reusable in different parts of their product) and as a narrative device in the case of the Queen – is an example of what we were looking for: bonds among CT, CS and AH. Their decision to generalise their game mechanic could be an interesting entry point for discussing abstraction, for instance, and that is the way we believe CT is meaningful: as a set of skills useful not only in technological domains, but also in others – such as AH, represented by narrative in this case. We can also employ the notion of procedurality to connect the narrative and CT thinking and operations: the narrative is procedural in its rule-governed nature, and in the conditionality produced by the player choices designed into it.

We acknowledge that these findings are limited due to the nature of a case study focused in a single game. In this sense, more extensive projects, including more participants and carried out through longer periods, could follow up this path in order to explore further this territory where CT, CS and AH can come together in learning experiences. In the same way, further research, less focused on the development of specific skills (e.g. programming) and more concentrated on how young people use these skills to produce and communicate meaning, is therefore needed if we want to establish and CT as a "formative skill".

This call for less focus on specific skills is by no means a way to argue in favour of scrapping programming from this kind of initiative. As discussed before, the establishment of a "common field" for these different areas depends on programming: since it is the orchestrating mode that shapes the experience, programming represents an important aspect when building bridges between these areas; moreover, as we have argued, a programming command can be close enough to a narrative statement depending on how it is constructed and read.

One last point that should be highlighted here is the role of CT in the contemporary world. CT is a common goal of educational systems nowadays, but it should not be pursued for its own sake; CT is not important in itself, but because it helps us to understand the world [11]. The possibility of producing an artefact in a more open way, less focused in teaching specific academic content, can play an important role in this process of connecting CT and modes of making meaning, especially because by doing so participants can firstly, feel more motivated to do it, and secondly, because it gives them a meaningful context to carry out this activity. In the case presented here, the game production offered a context for the students to play with concepts like abstraction and narrative meaning through digital media, which could later be further explored in different moments, either in formal or informal learning.

In this sense, game-design show itself as a good space for connecting these areas. This is not a new argument [21,50]; however, what we argue is not that game-making is a good opportunity to work with these areas because games depend on their subject content, but **because** games depend on these areas, they **demand** that people engage with CT, CS and AH and find means to integrate this diverse knowledge, empowering students to take part in the contemporary world.

Acknowledgements

The current research was approved by the relevant ethics committee at the UCL Institute of Education. Data was generated only after informed consent was obtained from both participants and their parents/guardians.

Data presented in this article was generated in AHRC project "Playing Beowulf: Gaming the Library", led by UCL Institute of Education, with the University of Sydney and the British Library (AHRC Reference: AH/M010201/1, 2015). The present article was supported by CAPES foundation, Ministry of Education—Brazil (Grant 1716/15-8).

References

[1] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, New York, 1980.
[2] I. Bogost, Procedural literacy: Problem solving with programming, systems, and play, Telemedium: J. Media Lit. 52 (2005) 32–36.
[3] B.A. Sheil, Teaching procedural literacy (presentation abstract), in: Proceedings of the ACM 1980 Annual Conference, ACM, New York, NY, USA, 1980, pp. 125–126. http://dx.doi.org/10.1145/800176.809944.
[4] A.A. DiSessa, Changing Minds: Computers, Learning, and Literacy, MIT Press, Cambridge, Mass., 2000.
[5] J.M. Wing, Computational thinking, Commun. ACM 49 (2006) 33–35. http://dx.doi.org/10.1145/1118178.1118215.
[6] J.J. Lu, G.H.L. Fletcher, Thinking about computational thinking, in: Proceedings of the 40th ACM Technical Symposium on Computer Science Education, ACM, New York, NY, USA, 2009, pp. 26–264. http://dx.doi.org/10.1145/1508865.1508959.
[7] L. Manovich, The Language of New Media, MIT Press, Cambridge, Mass., 2002.
[8] H. Jenkins, Convergence Culture: Where Old and New Media Collide, New York University Press, New York, 2006.
[9] D. Buckingham, Media Education: Literacy, Learning, and Contemporary Culture, Polity Press, Cambridge, UK, Malden, MA, 2003, Distributed in the USA by Blackwell Pub.
[10] A. Burn, J. Durran, Media Literacy in Schools: Practice, Production and Progression, Paul Chapman Publishing, London, 2007.
[11] Y.B. Kafai, From computational thinking to computational participation in K–12 education, Commun. ACM 59 (2016) 2–27. http://dx.doi.org/10.1145/2955114.
[12] J.M. Wing, Computational thinking and thinking about computing, Philos. Trans. A Math. Phys. Eng. Sci. 366 (2008) 371–3725. http://dx.doi.org/10.1098/rsta.2008.0118.
[13] UK Department for Education, The National Curriculum in England: Framework Document, 2013. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/335116/Master_final_national_curriculum_220714.pdf.
[14] A. Yadav, J. Good, J. Voogt, P. Fisser, Computational thinking as an emerging competence domain, in: M. Mulder (Ed.), Competence-Based Vocational and Professional Education, Springer International Publishing, 2017, pp. 105–1067. http://dx.doi.org/10.1007/978-3-319-41713-4_49.
[15] K. Howland, J. Good, Learning to communicate computationally with Flip: A bi-modal programming language for game creation, Comput. Educ. 80 (2015) 22–240. http://dx.doi.org/10.1016/j.compedu.2014.08.014.
[16] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, J.T. Korb, Computational thinking in elementary and secondary teacher education, Trans. Comput. Educ. 14 (2014) 5:1–5:16. http://dx.doi.org/10.1145/2576872.
[17] J. Jenson, M. Droumeva, Exploring media literacy and computational thinking: A game maker curriculum study, Electron. J. E-Learn. 14 (2016) 11–121.
[18] USA White House, Computer Science is for Everyone! Whitehouse.gov. 2013. https://obamawhitehouse.archives.gov/blog/2013/12/11/computer-science-everyone. (Accessed January 25, 2017).
[19] S. Grover, R. Pea, Computational thinking in K–12: A review of the state of the field, Educ. Res. 42 (2013) 38–43. http://dx.doi.org/10.3102/0013189X12463051.
[20] J. Voogt, P. Fisser, J. Good, P. Mishra, A. Yadav, Computational thinking in compulsory education: Towards an agenda for research and practice, EducInf Technol. 20 (2015) 715–728. http://dx.doi.org/10.1007/s10639-015-9412-6.
[21] V. Barr, C. Stephenson, Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads (2011) 48–54. http://dx.doi.org/10.1145/1929887.1929905.
[22] USA National Research Council, Report of a Workshop on the Pedagogical Aspects of Computational Thinking, The National Academies Press, Washington, DC, 2011. https://www.nap.edu/catalog/13170/report-of-a-workshop-on-the-pedagogical-aspects-of-computational-thinking.
[23] I. Lee, F. Martin, K. Apone, Integrating computational thinking across the K–8 curriculum, ACM Inroads 5 (2014) 64–71. http://dx.doi.org/10.1145/2684721.2684736.
[24] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith, L. Werner, Computational thinking for youth in practice, ACM Inroads 2 (2011) 32–37. http://dx.doi.org/10.1145/1929887.1929902.
[25] L. Mannila, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo, L. Rolandsson, A. Settle, Computational thinking in K-9 education, in: Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, ACM, New York, NY, USA, 2014, pp. 1–29. http://dx.doi.org/10.1145/2713609.2713610.
[26] E.R. Gee, K.M. Tran, Video game making and modding, in: B.J. Guzzetti, M. Lesley (Eds.), HandBook of Research on the Societal Impact of Digital Media, IGI Global, Hershey, PA, 2016, pp. 23–267.
[27] K. Salen, E. Zimmerman, Rules of Play: Game Design Fundamentals, MIT Press, Cambridge, Mass., 2004.
[28] J. Newman, Videogames, Routledge, London, New York, 2004.
[29] R. Khaled, A. Vasalou, Bridging serious games and participatory design, Int. J. Child-Comput. Interact. 2 (2014) 93–100. http://dx.doi.org/10.1016/j.ijcci.2014.03.001.
[30] K. Squire, Video Games and Learning: Teaching and Participatory Culture in the Digital Age, Teachers College Press, New York, 2011.
[31] I. Bogost, Unit Operations an Approach to Videogame Criticism, MIT Press, Cambridge, Mass., 2006. http://site.ebrary.com/id/10173643 (Accessed November 3, 2015).
[32] I. Bogost, Persuasive Games: The Expressive Power of Videogames, MIT Press, Cambridge, Mass., 2007.
[33] J.H. Murray, Hamlet on the Holodeck: The Future of Narrative in Cyberspace, MIT Press, Cambridge, MA, 1997.
[34] M. Sicart, The Ethics of Computer Games, MIT Press, Cambridge, Mass., 2009.
[35] J. Kramer, Is abstraction the key to computing? Commun. ACM 50 (2007) 37–42.
[36] Y. Mor, R. Noss, Programming as mathematical narrative, Int. J. Cont. Eng. Educ. Life-Long Learn. 18 (2008) 21–233.
[37] A.B. Lord, The Singer of Tales, Harvard University Press, Cambridge, 1960.
[38] M. Parry, Studies in the epic technique of oral verse-making. I: Homer and Homeric style', Harvard Stud. Class. Philology 41 (1930) 73–143.
[39] W.J. Ong, Orality and Literacy, second ed., Routledge, London, New York, 2002.
[40] A. Burn, G. Schott, Heavy hero or digital dummy? multimodal player–avatar relations in final fantasy 7, Vis. Commun. 3 (2004) 213–233. http://dx.doi.org/10.1177/147035704043041.
[41] A. Burn, J. Durran, Playing Shakespeare: Macbeth–Narrative, Drama, Game, Teaching English, 2013.
[42] G.R. Kress, Multimodality: A Social Semiotic Approach to Contemporary Communication, Routledge, London, New York, 2010.
[43] A. Burn, Games, films and media literacy: Frameworks for multimodal analysis, in: Researching New Literacies: Design, Theory, and Data in Sociocultural Investigation, Peter Lang, New York, NY, 2017.
[44] M. Walton, Semiotic Machines: Software in Discourse, Doctoral Thesis, University of Cape Town, 2008.
[45] D. Carr, D. Buckingham, A. Burn, G. Schott, Computer Games: Text, Narrative, and Play, Polity, Cambridge, Malden, MA, 2006.
[46] B. de Paula, Gamer-making: Discussing identities through game-making, Press Start. 3 (2016) 66–85.
[47] L. Werner, J. Denner, S. Campe, Using computer game programming to teach computational thinking skills, in: K. Schrier (Ed.), Learning, Education & Games, ETC Press, Pittsburgh, PA, 2014, pp. 37–54.
[48] G.R. Kress, T. Van Leeuwen, Reading Images: The Grammar of Visual Design, Routledge, London, New York, 2001.
[49] C. Pelletier, A. Burn, D. Buckingham, Game design as textual poaching: Media literacy, creativity and game-making, E-Learn. Digit. Media (2010) 90–107. http://dx.doi.org/10.2304/elea.2010.7.1.90.
[50] J. Good, K. Howland, Programming language natural language? Supporting the diverse computational activities of novice programmers, J. Vis. Lang. Comput. (2016). http://dx.doi.org/10.1016/j.jvlc.2016.10.008.
[51] Q. Burke, Y.B. Kafai, Decade of game making for learning: From tools to communities, in: M.C. Angelides, H. Agius (Eds.), Handbook of Digital Games, first ed., John Wiley & Sons, 2014, pp. 689–709.
[52] A. Burn, The Kineikonic Mode: Towards a Multimodal Approach to Moving Image Media, NCRM, London, UK, 2013. http://eprints.ncrm.ac.uk/3085/ (Accessed February 1, 2017).